

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**System and Method for Terminating Applications**

Inventor(s):  
Chee H. Chew

ATTORNEY'S DOCKET NO. MS1-319US

1 **CROSS-REFERENCE TO RELATED APPLICATIONS**

2 This application claims priority to U.S. Provisional Application No.  
3 60/110,003, filed November 24, 1998, entitled "System and Method for Weighted  
4 Application Termination", to Chee H. Chew.

5  
6 **TECHNICAL FIELD**

7 This invention relates to the termination of one or more applications  
8 running on a computer. More particularly, the invention relates to selecting a  
9 particular application for termination based on a priority value associated with  
10 each application running on a computer.

11  
12 **BACKGROUND OF THE INVENTION**

13 Laptop, handheld, and other portable computers or computing devices have  
14 increased in popularity as the devices have become smaller in size and less  
15 expensive. Additionally, improved operating speed and processing power of  
16 portable computers has increased their popularity. Many portable computers are  
17 capable of storing multiple application programs, such as address books, games,  
18 calculators, and the like. The application programs can be permanently installed  
19 in the portable computer during manufacture (e.g., on read-only memory (ROM)).  
20 Alternatively, the application programs may be installed by the user after  
21 purchasing the portable computer by downloading the application programs to the  
22 portable computer from a desktop computer.

23 Many of these small computers have limited physical resources, such as  
24 limited memory and limited processing capabilities. Thus, a typical portable  
25 computer may only be capable of executing a small number of application

1 programs simultaneously. In some systems, this problem is addressed by  
2 preventing simultaneous execution of multiple applications—by automatically  
3 closing any running applications before a new application is launched. While this  
4 is a workable solution, it imposes significant delays as a user switches between  
5 multiple applications. This type of delay can be frustrating to the user, especially  
6 if the applications require a significant amount of time to launch.

7 In other systems, multiple applications are allowed to execute concurrently.  
8 For example, a word processing application, a calendar application, and an address  
9 book application might be able to run simultaneously on a portable computer.  
10 However, the portable computer may not have sufficient resources remaining to  
11 launch an additional application program, such as a calculator. In this situation,  
12 one of the three running applications is terminated to reclaim system resources  
13 before launching the calculator application.

14 Typically, a small computer displays only one application at a time, rather  
15 than displaying multiple windows such as might be displayed on desktop  
16 computers. When a user of a small computer switches from one application to  
17 another, only the new application is displayed. Although other applications may  
18 continue running on the computer, they are not generally displayed until selected  
19 as the active application. Because of this, any non-visible application can be  
20 terminated without the notice of the user. When the user attempts to switch back  
21 to the terminated application, the terminated application is simply restarted at its  
22 previous state.

23 When an application must be terminated to reclaim resources, existing  
24 portable devices make an arbitrary decision as to which application will be  
25 terminated. In many cases, the terminated application will be the least recently

1 used application or the longest running application. Additional applications will  
2 continue to be terminated until enough system resources are available to initiate  
3 the newly selected application. Although this is usually successful, it does have  
4 disadvantages. One problem is that the application to be terminated might be in a  
5 modal state; i.e., waiting for user input. Attempting to close an application in a  
6 modal state may fail if the application refuses to terminate until the requested user  
7 input is received. A loss of data may result if the application in a modal state is  
8 forced to close before receiving the requested user input.

9 Another method of reclaiming system resources is to request that the user  
10 of a device select from listed applications to terminate. However, this can be  
11 confusing to the user, especially a user not familiar with computer systems. The  
12 user is not necessarily familiar with the characteristics of the applications and may  
13 not understand how different applications interact with one another. Users of  
14 portable devices often expect fast operation, for example when retrieving a  
15 meeting location from a calendar or a telephone number from an address book.  
16 These users may become frustrated if confronted with a question regarding  
17 terminating an application when they want to quickly retrieve data from the  
18 portable device.

19 The invention described below addresses these disadvantages and problems  
20 by allowing portable computers such as laptop computers, handheld computers,  
21 and palmtop computers to terminate one or more applications using a more  
22 "intelligent" selection system. In most cases, this greatly improves the operation  
23 and efficiency of the computer as perceived by the user, thereby increasing user  
24 satisfaction without requiring the addition of system resources such as memory.

25

## SUMMARY OF THE INVENTION

The invention provides a mechanism for terminating an application program to reclaim resources, such as memory or processor resources, in a computer. The invention is particularly useful in small computers, such as palmtop computers, having limited resources (e.g., limited memory and a limited number of concurrent processes). Rather than merely terminating the least recently used application or the longest running application, a computer in accordance with the invention selects the application to terminate based on various characteristics associated with executing applications. After analyzing these various characteristics, the "best" application to terminate is selected and terminated. The selection and termination of the application is performed automatically, without requiring any input from the user of the portable computer.

In accordance with the invention, a computer uses an "intelligent" selection process to determine which application to terminate. First, computer application programs executing on the computer are identified. Next, a priority value is assigned to each of the identified computer application programs. The priority value is determined based on multiple characteristics of the identified computer application programs. The computer application program with the smallest priority value is automatically terminated.

The characteristics associated with each of the computer application programs may indicate average launch times of the program, average memory usages of the program, the class or type of application, frequencies of usage, and the amount of data stored on the computer by the computer application program. The priority value for a particular computer application program can be

1 determined by adding together the values of all parameter values associated with  
2 the particular computer application program.

3 Another aspect of the invention includes determining whether the computer  
4 application program selected for termination is in a modal state in which it waits  
5 for a response from a user. If the selected application is in a modal state, then the  
6 system identifies a default response associated with the computer application  
7 program. The default response is provided to the computer application program.  
8 The process of identifying and providing default responses to the computer  
9 application program is repeated until the computer application program being  
10 terminated is in a non-modal state.

#### 11 12 **BRIEF DESCRIPTION OF THE DRAWINGS**

13 Fig. 1 illustrates an exemplary portable computer in accordance with the  
14 invention.

15 Fig. 2 is a block diagram illustrating pertinent components of a portable  
16 computer in accordance with the invention.

17 Fig. 3 is a flow diagram illustrating an exemplary procedure for  
18 determining priority values associated with applications programs on a portable  
19 computer.

20 Fig. 4 is a flow diagram illustrating an embodiment of a procedure for  
21 reclaiming system resources by terminating one or more applications.

22 Fig. 5 is a flow diagram illustrating an exemplary procedure for terminating  
23 applications in a modal state.  
24  
25

## DETAILED DESCRIPTION

Fig. 1 illustrates an exemplary portable computer 100 in accordance with the invention. Portable computer 100 can be any type of laptop, palmtop, handheld, or other computing device capable of executing one or more application programs. Portable computer 100 includes an LCD display 102 and an input mechanism that is typically something other than a full-size keyboard. Portable computer 100 includes several user input keys or buttons 104. The LCD display 102 is a touch-sensitive screen which, when used in conjunction with a stylus 106, allows a user to input information to portable computer 100. The stylus 106 is used to press the display at designated coordinates for user input. Buttons 104 provide another mechanism for user input. A particular portable computer may have any number of buttons for user input. Additionally, portable computer 100 may also include one or more slots or other receptacles (not shown) capable of receiving peripheral expansion cards.

In other embodiments of portable computer 100, the input mechanism might be a keypad, a track ball, a touch-sensitive pad, a miniaturized QWERTY keyboard, or the like. In other implementations, portable computer 100 may be implemented as a personal digital assistant (PDA), a personal organizer, a palmtop (or handheld) computer, a computerized notepad, or the like.

The application programs executed by portable computer 100 can be factory-installed by the manufacturer or may be contained on a peripheral storage card coupled to portable computer 100. Additionally, application programs can be downloaded into the memory of portable computer 100 from another computer, such as a desktop computer, using a communication link between the desktop computer and portable computer 100. The application programs executed by

1 portable computer 100 include, for example, word processing applications,  
2 spreadsheet applications, contact manager applications, and game applications.

3 Portable computer 100 has a limited amount of memory and processor  
4 resources available to execute application programs. If the user of portable  
5 computer 100 launches a new application program, it may be necessary to  
6 terminate a running application program to reclaim system resources, such as  
7 memory or processor resources. As discussed below, the present invention  
8 provides a mechanism for selecting a particular application running on portable  
9 computer 100 for termination based on a priority value associated with each  
10 application running on the portable computer. The selected application is  
11 terminated automatically without requiring any input by the user of the portable  
12 computer.

13 Although the present invention can be used with any type of computer  
14 system capable of executing application programs, it is particularly useful in  
15 computing devices with limited resources (e.g., limited memory or a limited  
16 number of concurrent processes). Throughout this specification, exemplary  
17 embodiments are described with reference to portable computer 100. However,  
18 similar procedures and components can be applied to any type of computing  
19 device.

20 Fig. 2 is a block diagram illustrating pertinent components of portable  
21 computer 100 in accordance with the invention. Portable computer 100 includes a  
22 built-in memory 200 and one or more memory expansion cards 202. A portion of  
23 built-in memory 200 is addressable memory for program execution, and the  
24 remaining portion is used to simulate secondary disk storage. The memory  
25 expansion cards 202 may contain permanently installed applications, such as



1 applications stored in a read-only memory (ROM), not shown. Additionally,  
2 memory expansion cards 202 may contain non-volatile memory for storing data or  
3 downloaded application programs, thereby supplementing built-in memory 200.  
4 Memory expansion cards 202 allow the user of portable computer 100 to  
5 customize the device by adding permanently installed application programs or  
6 adding memory for storing additional data and downloading additional application  
7 programs.

8 Memory 200 includes an operating system 220, one or more application  
9 programs 222, a registry 224, and an application termination module 226.  
10 Additionally, portable computer 100 has a processor 228, I/O components 230  
11 (including the display 102 and buttons 104 in Fig. 1), and a serial interface 232 for  
12 communicating with other computing devices (such as desktop computer 130 or  
13 another portable computer 100). In one embodiment, the various components in  
14 portable computer 100 communicate with one another over a bus 234. In one  
15 embodiment of portable computer 100, memory 200 is a non-volatile electronic  
16 memory such as a random access memory (RAM) with a battery back-up module,  
17 not shown. In an alternate embodiment, memory 200 is implemented using a flash  
18 memory device. Part of this memory 200 is addressable memory for program  
19 execution, and the remaining part is used to simulate secondary disk storage.

20 Operating system 220 executes on processor 228 from memory 200. In a  
21 particular embodiment of the invention, portable computer 100 runs the "Windows  
22 CE" operating system manufactured and distributed by Microsoft Corporation of  
23 Redmond, Washington. This operating system is particularly designed for small  
24 computing devices.  
25

1        Application programs 222 execute from memory 200 of portable computer  
2 100. The number of application programs 222 that can be simultaneously installed  
3 on portable computer 100 is a function of the portion of memory allocated to store  
4 application programs and the size of the application programs 222 currently  
5 installed.

6        The operating system 220 maintains registry 224. Registry 224 is a  
7 database that is implemented in various forms under different versions of the  
8 "Windows" operating systems. The registry contains information about  
9 applications stored on portable computer 100. Exemplary registry information  
10 includes user preferences and application configuration information. In  
11 accordance with the invention, various characteristics of different application  
12 programs are stored in the registry. When one or more applications need to be  
13 terminated to reclaim system resources, the registry is consulted to determine the  
14 "best" application to terminate, based in part upon the stored characteristics.

15        Application termination module 226 is a software component, and is part of  
16 the operating system 220 in the described embodiment. Application termination  
17 module 226 contains logic for determining which of the currently executing  
18 applications is the best candidate for termination, and then performs the steps  
19 necessary to terminate the application. The application to be terminated is  
20 identified by comparing priority values associated with the different executing  
21 applications. The priority values are calculated by application termination module  
22 226, based on different characteristics of the executing applications. As  
23 mentioned above, these characteristics are stored as parameter values in registry  
24 224. For example, the operating system defines the names of the parameters and  
25 the applications provide the value associated with the parameter. A parameter

1 such as "class" is defined by the operating system and may have an associated  
2 value of "1", "2", or "3". A value of "1" indicates that the application is a  
3 personal information manager, a value of "2" indicates that the application is a  
4 game, and a value of "3" indicates that the application is a utility. Another  
5 parameter such as "memory" has an associated value that indicates the typical  
6 memory usage of the application.

7 Fig. 3 is a flow diagram illustrating an exemplary procedure for  
8 determining priority values associated with application programs on a portable  
9 computer. Step 250 comprises identifying each application program on the  
10 portable computer. Step 252 comprises identifying different characteristics of  
11 each application program on the portable computer. Such characteristics are  
12 preferably represented by parameter values stored in registry 224. Any number of  
13 parameters can be used to identify characteristics of the particular application  
14 programs. Typically, the parameter values are set by the developer of each  
15 application program. Each of the parameters is related to a characteristic that is  
16 useful in determining which application will be terminated. The parameters  
17 associated with a particular application are registered with the operating system by  
18 calling an API function provided by operating system 220. The operating system,  
19 in turn, stores the parameters in registry 224. Typically, the parameters are  
20 provided to the operating system by the installation program, when the application  
21 is initially installed on the computer. Alternatively, the installation program might  
22 copy the parameters directly to the registry or to some other memory location that  
23 is accessible to application termination module 226.

24 Any number of parameters can be used to describe the characteristics of a  
25 particular application. For example, an average launch time parameter indicates

1 the time required to restart the application program if it is terminated. An  
2 application with a short launch time is more likely to be terminated than an  
3 application with a long launch time. An average memory usage parameter  
4 indicates the typical amount of memory used by the application program when  
5 executing. Applications that use a large amount of memory may be terminated  
6 before applications that use less memory.

7 An application class parameter indicates the application's type, such as a  
8 game, utility, or personal information manager (PIM). The application class can  
9 be useful in determining which application to terminate. For example, if three  
10 games are running, one of the games may be terminated instead of terminating a  
11 word processor or PIM. A frequency of usage parameter indicates how often the  
12 application program is used or accessed by the user of the portable computer. An  
13 infrequently used application program is more likely to be terminated than a  
14 frequently used application program. An amount of data stored parameter  
15 identifies the quantity of data a user has stored using a particular application  
16 program. If a user has stored a significant amount of data using a particular  
17 application program, that application program is less likely to be terminated.

18 After identifying parameter values in step 252, the procedure of Fig. 3  
19 continues to step 254, which comprises calculating a priority value associated with  
20 each application program on the portable computer. In one embodiment, step 254  
21 of Fig. 3 calculates the priority value associated with a particular application  
22 program by adding the values of all parameters associated with that particular  
23 application program. For example, if five parameters are associated with each  
24 application program on a portable computer, the priority value is calculated using  
25 the following formula, where  $P_x$  represents a parameter:

$$\text{Priority Value} = P_1 + P_2 + P_3 + P_4 + P_5$$

Typically, the parameters are designed so that higher values indicate less likelihood of an application being terminated. For example, an application that uses a small amount of memory will have a larger memory usage parameter value than an application that uses a large amount of memory. When all of the parameters are added together to determine the priority value, the application with the smallest associated priority value is terminated. Using this formula for calculating the priority value, several parameters are considered when selecting an application to terminate, rather than relying on a single parameter.

In an alternate embodiment, a weighting factor may be applied to each parameter indicating the weight or importance of each parameter. For example, if five parameters are associated with each application program on a portable computer, an important parameter may receive a weighting factor of five and an unimportant parameter may receive a weighting factor of one. Using this alternate embodiment, the priority value can be calculated using the following formula, where  $P_x$  represents a parameter and  $W_x$  represents a weighting factor associated with the parameter:

$$\text{Priority Value} = P_1W_1 + P_2W_2 + P_3W_3 + P_4W_4 + P_5W_5$$

After calculating the priority values in step 254, the procedure of Fig. 3 continues to step 256, which comprises storing the priority values in a register or other storage mechanism. By storing these priority values in a register within the

1 portable computer, the portable computer is able to quickly retrieve the priority  
2 values when it has determined that an application needs to be terminated.

3 Table 1 below illustrates five parameters associated with each of six  
4 application programs. Although Table 1 contains five specific parameters, the  
5 present invention can use any number of parameters (including parameters not  
6 listed in Table 1) to determine priority values associated with application  
7 programs. Each parameter value is assigned by each executing application. For  
8 example, the value associated with "Average Launch Time" parameter is one per  
9 every 100ms, such that an application with an average launch time of 600ms has  
10 an associated value of six. The value associated with Average Memory Usage  
11 indicates the average number of kilobytes of memory used by the application. The  
12 application class has an associated number that indicates the class or type of  
13 application. For example, 1 = a personal information manager (PIM), 2 = a game,  
14 3 = a utility, and 4 = a productivity application). The Frequency of Usage  
15 parameter is determined by the operating system based on monitoring the usage of  
16 the application. The value associated with the Frequency of Usage parameter may  
17 indicate the percentage of total operating time during which the application was  
18 executing. For example, if a particular application is executing for a few minutes  
19 during each hour that the portable computer is powered on, that application is  
20 assigned a Frequency of Usage value of 1. However, an application that runs for  
21 the majority of the time that the portable computer is powered on (e.g., 70% of the  
22 time), the application is assigned a Frequency of Usage value of 7. The Amount  
23 of Data Stored parameter indicates the data storage space used by the application,  
24 measured in kilobytes.

Application Program	Average Launch Time	Average Memory Usage	Application Class	Frequency of Usage	Amount of Data Stored
Calculator	2	5	3	1	0
Word Processor	10	20	4	4	15
Spreadsheet	12	30	4	3	12
Calendar	8	20	1	7	22
Address Book	5	10	1	4	18
Solitaire	3	10	2	1	1

Table 1

Certain parameter values (such as application class) can be assigned by the developer of the application program. However, other parameter values (such as frequency of usage and amount of data stored) can only be determined after the application program has been executed for some period of time. These parameters (frequency of usage and amount of data stored) are dependent on the user's habits and program selection rather than being dependent on an inherent characteristic of the application program. The operation of these programs must be monitored to collect data that will define these parameter values.

The procedure described above with respect to Fig. 3 is used by application termination module 226 to calculate a priority value for all application programs on the portable device, regardless of whether the application is currently being executed. In another embodiment of the invention, the procedure of Fig. 3 calculates and stores a priority value for all application programs currently being executed, but does not calculate priority values for application programs that are not running. This alternate embodiment typically reduces the time required to calculate priority values because it ignores applications that are not running.

1 These applications can be ignored because they will not be considered for  
2 termination until they are executing. This alternate embodiment calculates and  
3 stores the priority value associated with each newly executed application when the  
4 application is launched. Thus, priority values are calculated on an as-needed basis  
5 in this embodiment.

6 Similarly, in one embodiment of the invention, Table 1 above contains  
7 parameters for every application program stored on the portable computer. In  
8 another embodiment, Table 1 contains parameters for running applications, but  
9 does not store parameters for non-executing programs. This second embodiment  
10 requires storage of fewer parameters and is more efficient because it does not store  
11 information for non-running applications, which will not require termination.

12 In another embodiment of the invention, the priority values are calculated  
13 (or recalculated) at the time an application needs to be terminated. Various  
14 conditions are considered when assigning or modifying priority values associated  
15 with the application programs executing on the portable computer. For example, if  
16 multiple games are running on the portable computer, then the older game is  
17 assigned a lower priority value (a stronger candidate for termination) because two  
18 games are not typically played simultaneously. If the maximum number of  
19 concurrent processes is reached, but available memory remains relatively high,  
20 then fast-loading applications are assigned a lower priority value because they can  
21 be quickly re-loaded if necessary. If memory in the portable computer is low, then  
22 applications that use larger amounts of memory are assigned a lower priority  
23 value. Applications that have not been accessed during the last user session are  
24 assigned a lower priority value because of the lack of use. If a particular portable  
25



1 computer is a PIM, then applications that are classified as non-PIM programs are  
2 assigned a lower priority value, thereby giving a preference to PIM applications.

3 In an exemplary embodiment of the invention, the application with the  
4 highest priority value is terminated. Initially, the priority value assigned to each  
5 application is zero. In this example, preference is given to PIM applications by  
6 adding 100 to the priority value of each non-PIM application. If more than one  
7 game is executing simultaneously, the priority value of the first game is not  
8 changed, but the priority values of all other games executing simultaneously are  
9 increased by 500. For each user session that an application is not used (i.e., the  
10 application is executing, but not accessed by the user), the priority value of the  
11 application is increased by 250. Thus, the longer an application is not accessed,  
12 the greater the possibility that the application will be selected for termination. If  
13 the system is low on memory, applications with high memory usage have their  
14 priority value increased relative to their memory usage. For example, an  
15 application with very high memory usage has its priority value increased by 500  
16 while an application with moderate memory usage has its priority value increased  
17 by 250.

18 Continuing with the exemplary embodiment, if an application has a slow  
19 launch time, it is desirable to keep this application executing to avoid a long delay  
20 in launching the application after a termination. An application with a slow launch  
21 time has its priority value reduced by 100 while an application with a fast launch  
22 time has its priority value increased by 100. In certain situations, an application  
23 with a particular characteristic may be terminated even though it does not have the  
24 highest priority value. For example, if the computer is reaching the maximum  
25

1 number of concurrent processes, an application with a fast launch time may be  
2 terminated even if other applications have higher priority values.

3 Fig. 4 is a flow diagram illustrating an embodiment of a procedure for  
4 reclaiming system resources by terminating one or more applications. Step 270  
5 comprises receiving a request to launch a new application program. Step 272  
6 comprises checking the available resources (e.g., available memory and processing  
7 resources) in the portable computer. Step 274 comprises checking the resources  
8 required to execute the new application program. This information can be  
9 obtained, for example, from the parameters associated with the new application.  
10 These parameters may indicate the typical memory and other resources required to  
11 execute the application on the portable computer.

12 Step 276 determines whether the available resources in the portable  
13 computer are sufficient to handle the new application program. This  
14 determination is accomplished by comparing the available resources in the  
15 portable computer with the expected resources required to execute the new  
16 application (as identified by the parameters associated with the new application).  
17 The current resources in the portable computer can be determined by polling for  
18 available memory or the number of processes currently running. If the available  
19 resources are sufficient to handle the new application, then the procedure branches  
20 to step 278, which comprises executing the requested application program.

21 If the available resources are not sufficient to handle the new application,  
22 then the procedure branches from step 276 to step 280, which comprises  
23 identifying the priority values associated with each application running on the  
24 portable computer. These priority values can be retrieved from the registry or  
25 other storage device used in step 256 of Fig. 3. Step 282 of Fig. 4 terminates the

1 application with the lowest priority value. The procedure then returns to step 276  
2 to determine whether the currently available resources (after terminating the  
3 application in step 282) are sufficient to handle the new application program. The  
4 procedure continues terminating applications until the available resources are  
5 sufficient to handle the new application program. Additionally, if the resources  
6 are sufficient, but the program scheduler in the portable computer is switching  
7 continuously (e.g., thrashing), then step 276 may branch to 280 to terminate an  
8 application, even though the current resources are sufficient. Terminating an  
9 application prior to launching the new application may help reduce the continuous  
10 switching between processes.

11 The procedure illustrated in Fig. 4 is implemented without requiring any  
12 input from the user of the portable computer. Thus, the user of the portable  
13 computer selects the new application program to launch by pressing the  
14 appropriate button or touch-screen location on the portable computer. The  
15 procedure of Fig. 4 is then performed without any further intervention by the user  
16 of the portable computer. The user does not know which, if any, applications were  
17 terminated. After selecting the new application program to launch, the next item  
18 displayed to the user is the information associated with the new application  
19 program.

20 The procedure of Fig. 4 is initiated in response to a request to launch a new  
21 application program. In another embodiment, a portion of the procedure of Fig. 4  
22 is executed when available system resources fall below a particular threshold  
23 value. For example, if the available memory falls below a predetermined  
24 threshold, then steps 280 and 282 in Fig. 4 are executed to terminate an  
25 application, thereby reclaiming system resources. In another example, steps 280

1 and 282 are performed when the number of concurrent processes executing on the  
2 portable computer exceeds a threshold. Thus, it is not necessary to wait for a  
3 request to launch a new application program before terminating an application.  
4 An application can be terminated any time that available system resources fall  
5 below a particular threshold.

6 In one embodiment of the invention, a particular computing device may  
7 contain one or more "core" applications that are never candidates for termination.  
8 A core application may be required to execute other applications or may represent  
9 the most frequently used applications in the computing device. The core  
10 application may be selected by the user of the computing device or determined by  
11 the computing device itself based on historical application usage or predetermined  
12 by the developer of the computing device. For example, if the computing device  
13 is primarily used to maintain a calendar of meetings and events, then the calendar  
14 application may be designated as a core application because it should always be  
15 readily available to the user of the computing device. In other embodiments of the  
16 invention, there may be no core application, such that any application is a valid  
17 candidate for termination.

18 To avoid selecting a core application for termination, the computing device  
19 may assign a unique code to the application's priority indicating that the  
20 application is a core application. Alternatively, a very high priority may be  
21 assigned to the core application such that it will always have the highest priority  
22 value and, therefore, not be selected for termination.

23 Fig. 5 is a flow diagram illustrating an exemplary procedure for terminating  
24 applications in a modal state. An application in a modal state has requested an  
25 input from the user of the application, for example in the form of a dialog box. In

1 the modal state, the application is waiting for the user's response. At step 290, an  
2 application is selected for termination. Step 292 determines whether the  
3 application selected for termination is in a modal state. If the application is not in a  
4 modal state, then the procedure branches to step 298, which terminates the  
5 application selected for termination.

6 Terminating an application in the modal state may result in the loss or  
7 corruption of data stored by the application. Therefore, if the application selected  
8 for termination is in a modal state, then the procedure continues to step 294 which  
9 determines a default response associated with the application selected for  
10 termination. One or more default responses may be associated with the  
11 application depending on the number of different inputs the application may  
12 request from the user. Step 296 generates a default response and communicates  
13 the default response to the application, thereby providing the input requested by  
14 the application. The procedure then returns to step 298 to be certain that the  
15 default response removed the application from its modal state. If the application is  
16 no longer in a modal state, then the procedure branches to step 298, which  
17 terminates the application. However, if the application remains in a modal state  
18 (e.g., the application requested additional input from the user), then the procedure  
19 branches to step 294 to identify and generate another default response. The  
20 procedure continues providing default responses to the application until the  
21 application is in a non-modal state, at which point the application is terminated.

22 The default responses may be generated by the application itself in  
23 response to an operating system request to terminate. Alternatively, the responses  
24 can be generated using an application programming interface (API) or similar  
25 mechanism that facilitates the creation of default responses. In another

1 embodiment, each application provides one or more default responses to the  
2 operating system when the application is launched. If the operating system  
3 determines that an application in a modal state should be terminated, then the  
4 operating system provides the appropriate default response to the application (i.e.,  
5 the default response received from the application when launched).

6 When generating a default response for an application in a modal state, if a  
7 termination command is received and the associated application has one or more  
8 open dialog boxes, a default response is provided to each open dialog box. If a  
9 termination command is received and, later, a request to open a dialog box for the  
10 application is received before the application is terminated, then the dialog box is  
11 not opened. Instead, the default response that would have been generated if the  
12 dialog box was open is returned to the application to allow the application to be  
13 terminated without loss or corruption of data.

14 The described system provides a significant advantage over the prior art.  
15 Specifically, it allows applications to be terminated in a way that is least intrusive  
16 to the user's actual interaction with a computer. By utilizing appropriate  
17 parameters, the perceived responsiveness of the computer will be greatly  
18 increased, thereby increasing the usefulness of the computer and in many cases  
19 reducing the amount of hardware resources that will be required in the computer.

20 Although the invention has been described in language specific to structural  
21 features and/or methodological steps, it is to be understood that the invention  
22 defined in the appended claims is not necessarily limited to the specific features or  
23 steps described. Rather, the specific features and steps are disclosed as preferred  
24 forms of implementing the claimed invention.  
25